Unidad 3



Recuperación de datos

mediante Oracle 11g

Apuntes realizados para la asignatura de FP Grado Superior:

Administración de Bases de Datos

del ciclo Administración de Sistemas Informáticos en Red

Autor: Jorge Sánchez Asenjo (www.jorgesanchez.net)

Versión del documento: 2.1, Año 2013



índice

(3) recuperación de datos

(3.1) introducción

(3.1.1) introducción

Indudablemente una de las tareas más importantes en la administración de las bases de datos es la de realizar copia y restauración de la base de datos.

Lo ideal es que esta tarea permita recuperar absolutamente todos los datos en caso de catástrofe; pero no siempre es posible. Un buen administrador deberá establecer métodos de trabajo que permitan asegurar la base de datos de modo que las copias habituales y la política de la base de datos aseguren al máximo tanto los datos como las estructuras necesarias para que Oracle funcione con normalidad tras el suceso que provocó la necesidad de la copia.

La operación de restauración nos permitirá recuperar aquella información dañada o perdida. Oracle proporciona numerosas posibilidades para esta tarea (hay manuales completos de Oracle sólo dedicados a explicar las múltiples posibilidades de realizarla), en estos apuntes se comentan las copias y restauraciones de usuario.

(3.1.2) causas de pérdida de datos

Hay numerosas razones que pueden provocar el desagradable efecto de perder datos:

- **Errores de usuario**. Es curioso pero la mayoría de los problemas de pérdida de datos tienen que ver con el propio usuario. Es decir, ocurren porque el propio usuario o usuaria elimina la información.

  Las instrucciones DML (**INSERT**, **DELETE** o **UPDATE** de SQL) de manipulación de datos son reversibles en las bases de datos transicionales (especialmente si cumplen la norma **ACID**), pero no los son las instrucciones DDL (como la temible **DROP TABLE** por ejemplo).

  Además cuando la transacción se acepta (con un **COMMIT** o realizando una instrucción DDL, por ejemplo), entonces pasa a ser definitiva. Si la transacción había borrado datos éstos se habrán perdido.

  Son los fallos más complicados de arreglar porque responden a una voluntad de borrar definitivamente los datos, la única posibilidad de recuperar los datos estará en la existencia de al menos una copia de los mismos.

- **Errores en la ejecución de instrucciones**. Ocurren cuando una instrucción no pudo ejecutarse correctamente porque no cumple una determinada restricción

y no se lleva a cabo y los datos no llegan a grabarse. Si la instrucción era compleja y había provocado empezar a volcar los datos hasta llegar a uno que no cumple la condición, entonces la instrucción se detiene pudiendo dejar datos incoherentes al no finalizar la instrucción.

Normalmente ante un error en una instrucción, una base de datos como Oracle anula la misma y si era parte de una transacción las demás siguen en el mismo estado: es decir sólo se anula la del error.

- **Errores en la programación de aplicaciones**. En realidad es un problema parecido al anterior. Un caso habitual ocurre cuando resulta que una aplicación permite a varios usuarios trabajar con los mismos datos concurrentemente habiendo posibilidad de un **abrazo mortal** (dos sesiones que se bloquean mutuamente y cuelgan la base de datos) y los datos que intentan grabar no se graben jamás y, por lo tanto, se pierdan.

  Hay otros numerosos ejemplos de estos problemas, bucles infinitos que escriben y escriben datos sin control hasta llenar los tablespace, problemas con los permisos de usuario al escribir, mal control de tablas mutantes en PL/SQL,… Casi todo se detecta mediante los monitores de la base de datos y se resuelve programando correctamente las aplicaciones.

- **Errores de red**. Ocurre cuando la red pierde la conexión bien por problemas de hardware (el cable se desconecta, un aparato de red deja de funcionar,…) o de software (el listener se bloquea). Se puede evitar el problema disponiendo de elementos redundantes de red en el servidor: dos interfaces de red conectadas a dos estructuras distintas de red.

  En el caso de que el cliente pierda la conexión con la red y la sesión no pueda cerrarse con normalidad, Oracle anulará la transacción iniciada (si la había)

- **Fallos de disco**. Ocurren cuando el disco se deteriora y se estropea, bien impidiendo escribir en él (o en el caso más grave incluso impidiendo recuperar la información en él) o bien cuando datos críticos de la base de datos se corrompen por el mal estado del disco provocando un mal funcionamiento de la base de datos e incluso la pérdida de datos.

  Por ello es una temeridad no utilizar discos RAID que permitan duplicar los datos para el caso de que falle un disco la base de datos continúe trabajando sin problema y simplemente al reemplazar el disco deteriorado volvamos a tener la base de datos asegurada.

  Más grave es el problema de que un usuario del sistema operativo en el que está instalada la base de datos tenga privilegios suficientes para borrar archivos de disco de la base de datos y lo haga. En ese caso la situación es más dramática porque no puede ser controlada por el sistema gestor. Una vez más dependeremos de nuestras copias ya que realmente es un fallo de usuario.

- **Errores críticos**. Son fallos que provocan la caída de la instancia de la base de datos bien por un mal funcionamiento del software de sistema operativo (a veces también causado por fallos en disco) o por problemas en el hardware (desde problemas en la memoria RAM hasta que se queme la placa base del servidor) que causan la parada de la base de datos y, probablemente pérdidas de datos durante el acceso a la misma (además de posibilidad de incoherencia en la estructura de la base de datos que cause su mal funcionamiento).

  Podemos simular este fallo mediante el uso del comando **SHUTDOWN ABORT** que hace que se apague de golpe la base de datos. Como la información primero se almacena en memoria RAM y luego en disco, perderemos los datos

en memoria y además podemos haber empezado a almacenar transacciones y dejarlas a medio camino provocando una base de datos corrupta (con datos incoherentes) que incluso podría negarse a ponerse en funcionamiento.

Hay que recordar que los datos tardan en grabarse desde el COMMIT hasta la grabación en los archivos de disco. Realmente la grabación de los datos la realiza **LGWR** en los archivos redo log. Sólo cuando el proceso **DBWn** recibe la notificación **CHECKPOINT** se graban físicamente en los archivos de datos. De otro modo se multiplica la posibilidad de que los datos puedan quedar corruptos por un fallo grave. Aun así puede ocurrir y la situación sería grave.

El hecho de que Oracle grabe casi a tiempo real mediante LGWR, permite que en caso de fallo grave a través de los archivos Redo podamos recuperar correctamente la instancia de Oracle.

(3.1.3) tipos de copias de seguridad

- **Realizadas en frío**. Con la base de datos apagada y, por lo tanto, sin permitir el acceso a los usuarios. Permite una copia exacta, pero obviamente en grandes sistemas en producción no se contempla siquiera, porque la base de datos tiene que estar permanentemente en funcionamiento.
- **Realizadas en caliente**. Permiten realizar la copia mientras la base de datos sigue en ejecución. Son más complejas, pero son las más utilizadas sistemas de alta disponibilidad.
- **Lógicas**. Se realizan en caliente y son las que permiten exportar o importar datos concretos de la base de datos.
- **Físicas**. Pueden ser en frío o en caliente, realizan una copia exacta de la base de datos.

(3.2) almacenamiento de datos en Oracle

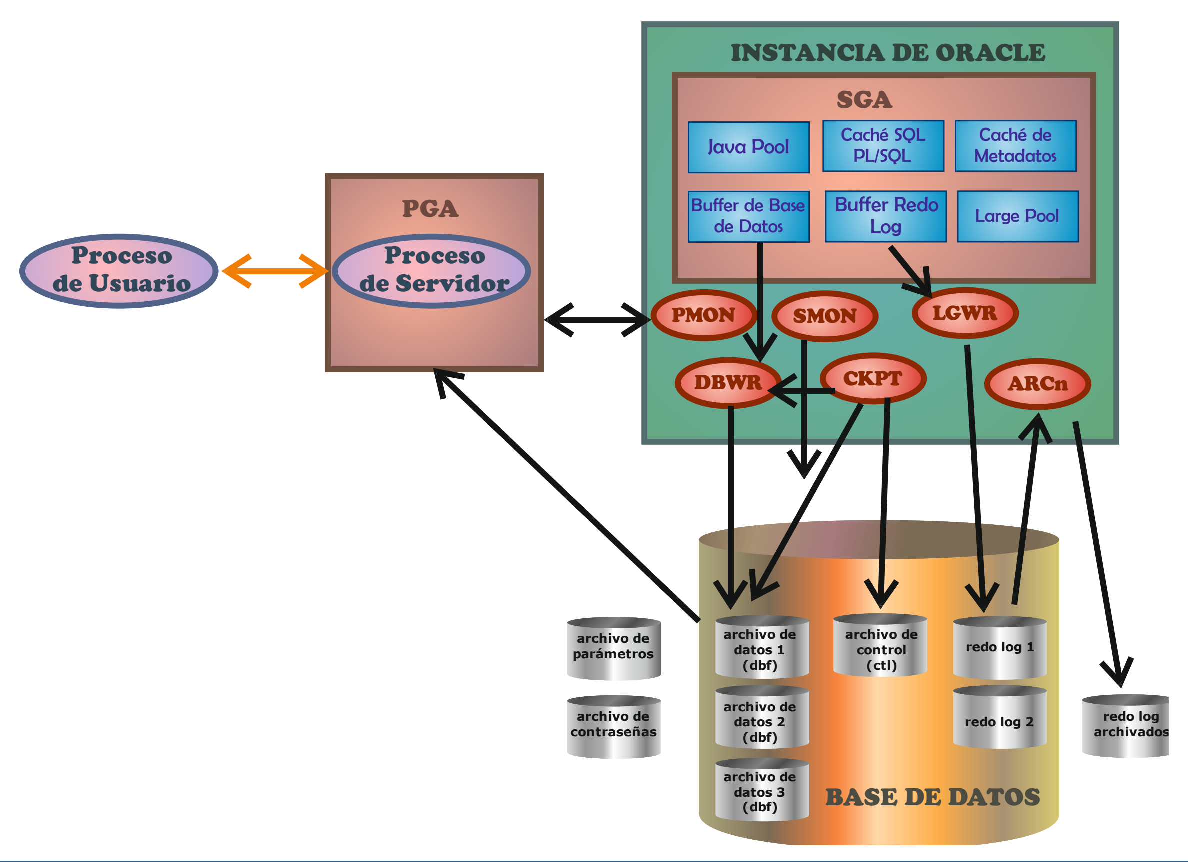
(3.2.1) repaso a la arquitectura de Oracle

En la unidad 1, se explicó el funcionamiento del gráfico a grandes rasgos, sin embargo ahora necesitamos profundizar en dicho funcionamiento para comprender la forma que tiene Oracle de almacenar los datos. Recordamos algunos conceptos:

- **tablespaces**. Estructura lógica utilizada para almacenar objetos de la base de datos. Se corresponde con uno o más archivos de datos en disco. Se pueden consultar mediante la vista **USER_TABLESPACES** que muestra la información sobre los tablespaces actuales.
- **segmentos**. Cada uno se corresponde común objeto de la base de datos. Hay cinco tipos principales de posibles segmentos:
  - **Segmentos de tabla**. Que almacenan los datos de una tabla concreta
  - **Segmentos de índice**. Que permiten acelerar las operaciones de búsqueda y ordenación de datos.
  - **Segmentos *undo* y de *rollback***. Que permiten organizar los datos durante las transacciones para que la información sea coherente durante la transacción y para poder anular la transacción manteniendo íntegra la coherencia.

- **Segmentos temporales**. Necesarios cuando se necesita almacenamiento extra para poder concluir una operación que provoca consulta sobre gran cantidad de datos. Al finalizar la operación, el segmento se elimina.
- **Segmentos de anulación (bootstrap)**. Están sólo en el tablespace SYSTEM y contienen el diccionario de datos fundamental.

- **Extensiones**. División lógica de los segmentos. Cuando se llena una extensión, el sistema gestor de base de datos crea la siguiente y reserva ese espacio en disco.
- **Bloques**. Cada uno de los elementos en los que se divide una extensión.



(3) Ilustración 1, Arquitectura interna de Oracle

vistas del diccionario de datos relacionadas con los tablespaces

- **USER_TABLESPACES**. Información sobre las tablespaces relacionadas con los datos del usuario actual.
- **DBA_TABLESPACES.** Información sobre los tablespaces globales. Sólo disponible esta vista para usuarios con permisos de DBA.
- **USER_SEGMENTS**. Información sobre los segmentos de datos relacionados con el usuario actual.
- **DBA_SEGMENTS.** Información sobre los segmentos de todos los usuarios/as.
- **DBA_DATAFILES**. Información sobre los archivos de datos. Sólo disponible para usuarios de tipo DBA.

- **USER_EXTENTS**. Información sobre las extensiones que pertenecen al usuario actual. **DBA_EXTENTS** disponible para los administradores, permite mostrar información sobre todas las extensiones de la base de datos

almacenamiento de datos en Oracle

proceso DBW*n*

En la primera unidad hablamos de este proceso, encargado de escribir definitivamente los datos en disco cuando el proceso **CKPT** le avisa de la llegada de un checkpoint, que es un instante de tiempo definido que cuando se cumple obliga a actuar al proceso DBW.

Mejor dicho, son los procesos (en plural) **DBW*n*** porque en realidad puede haber varios (hasta 20) dependiendo de la potencia del servidor. Así habrá proceso DBW0, DBW1, etc.

Hay dos formas de actuar de DBW, la diferencia está en el tipo de checkpoint que le llega. Hay un **checkpoint incremental** y hay otro **total**. En realidad cuando nos referimos a un checkpoint, es el segundo; pero el primer tipo es el que más ocurre.

Cuando los usuarios modifican los datos de la base de datos, ocupan búferes de datos en el área de la SGA dedicada a ello. De tal manera que se van abriendo búferes continuamente. Estos búferes se van llenando con los datos que los usuarios necesitan añadir a la base de datos, de modo que a esos búferes se les llama **búferes sucios**. Por el contrario los búferes que están actualmente disponibles se les llama **búferes libres**.

La cuestión es que puede haber miles y miles de búferes ocupados para ser guardados en los archivos de datos, lo que supone que esa sea una tarea larga y, por lo tanto, peligrosa porque durante todo el tiempo de escritura, un error crítico podría dejar la base de datos corrupta y no funcionar.

Los **checkpoints** incrementales sólo escriben unos cuantos búferes sucios y les vacían dejándoles libres, así el tiempo de grabación es más lento. Sólo cuando llega el checkpoint de verdad, el total, se graba todos los búferes sucios. Esto ocurre muy de vez en cuando por lo que el riesgo es menor.

La cuestión es que por este modo de trabajo los archivos de datos nunca están al día y dependeremos de los archivos redo log para poder al día los datos en caso de errro crítico.

DBW actúa si ocurre una de estas circunstancias (las tres primeras se corresponden a un checkpoint incremental y sólo la última es un checkpoint total):

(1) **No hay búferes libres** en el búfer de datos de la SGA para seguir almacenando información. Si esto ocurre Oracle escribe bloques sucios en disco para poder liberar la ocupación de la memoria y disponer de más búferes libres.

(2) Hay **demasiados bloques sucios**. En ese caso se escriben unos cuantos bloques sucios en disco para liberar espacio. Es un caso previo al anterior que se basa en una medida de ocupación preventiva para no esperar a que no haya búferes libres, que sería más dramático.

(3) **Se cumplió el tiempo de tres segundos de espera**. En una base de datos ocupada, este evento no se cumple jamás. Ocurre cuando el sistema está sin hacer nada durante tres segundos. En ese caso se escriben unos cuantos bloques sucios. Si la base de datos sigue desocupada durante mucho tiempo, acabará volcando poco a poco todos los bloques de los búferes.

(4) La aparición de un **checkpoint** (lo que hemos denominado un checkpoint total). Esto provoca escribir absolutamente todos los bloques sucios. Es el momento de máximo trabajo en disco y, por lo tanto, el momento más crítico porque en un checkpoint se pueden tener que grabar miles de datos. Por ello Oracle inicialmente pone el parámetro que controla el tiempo de aparición de un checkpoint a cero (que es lo mismo que tiempo infinito).

Hay cuatro eventos que provocan un checkpoint (es decir que el proceso **CKPT** se lance y avise a DBWR):

i. Que se hayan escrito en los archivos Redo Log el número de bloques marcados en el parámetro **LOG_CHECKPOINT_INTERVAL**. Normalmente este parámetro está configurado con el valor cero lo que significa que jamás se cumplirá. Podemos indicarle otro valor (con **ALTER SYSTEM SET**).

ii. Que haya transcurrido el tiempo marcado en **LOG_CHECKPOINT_TIMEOUT** (suele ser de tres minutos) desde la última vez que ocurrió un checkpoint.

iii. Cuando se cierra la base de datos (excepto con SHUTDOWN ALERT).

iv. Cuando se invoca al comando **ALTER SYSTEM CHECKPOINT**

Hay tres parámetros de sistema relacionados con los **checkpoints**:

- **LOG_CHECKPOINT_INTERVAL**. Que especifica el número de bloques máximo que se escriben en los redo log antes de que ocurra un checkpoint.
- **LOG_CHECKPOINT_TIMEOUT**. Intervalo en segundos (por defecto 180) máximo tras el cual se lanzará un checkpoint.
- **LOG_CHECKPOINTS_TO_ALERT**. Vale TRUE o FALSE (por defecto FALSE) para indicar si cada vez que ocurra un checkpoint, se producirá una entrada informando del mismo en el log de alertas (**alert log**). El archivo de alertas está situado en el directorio indicado en el parámetro **BACKGROUND_DUMP_DEST**.

redo log, proceso LGWR

funcionamiento de los archivos redo log

Los redo log son dos o más archivos que almacenan los cambios que se van registrando en la base de datos. Graban las instrucciones ejecutadas y los datos necesarios para volver a realizarlas. Son al menos dos, para asegurar que uno siempre recoge los cambios de la base de datos mientras el otro queda almacenado en disco.

Su funcionamiento es circular. Es decir se graba la información en uno y cuando se llena se pasa al siguiente. Tras llenar el último, comenzaremos de nuevo por el primero.

La finalidad de los redo log es facilitar la tarea de recuperar información de la base de datos. El funcionamiento consiste en lo siguiente:

(1) El proceso **LGWR** se encarga de mantener los archivos redo log al día. Copia las instrucciones desde el proceso servidor que recibe las instrucciones al espacio de buffer para redo logs.

(2) Desde el búfer se vuelcan al disco cuando son definitivos (con COMMIT por ejemplo), de ahí se colocan en el primer archivo redo log. Se van volcando los datos continuamente a medida que los usuarios les van produciendo (o borrando o modificando).

(3) Cuando llenamos el archivo se produce el evento **log switch**, entonces los siguientes datos pasan al siguiente archivo redo log. Con éste ocurrirá lo mismo y así sucesivamente con cada archivo del que dispongamos.

(4) En cada cambio de archivo, se genera un **número de secuencia** que indica los cambios de archivos que se han realizado (lo que es lo mismo, los eventos **log switch** que han ocurrido), así podemos tener ahora el número de secuencia 12 indicando que hemos hecho 12 cambios de archivo (o sea, han ocurrido **12 log switch**).

(5) Cuando ya hemos ocupado todos los archivos, al terminar de llenar el último se produce el consiguiente log switch y como no hay más archivos disponibles, grabará los siguientes datos en el primer archivo sobrescribiendo los que ya existieran (y perdiendo esos datos). El número de secuencia seguirá incrementándose, no volverá a empezar de nuevo.

(6) Todo este proceso sigue continuamente

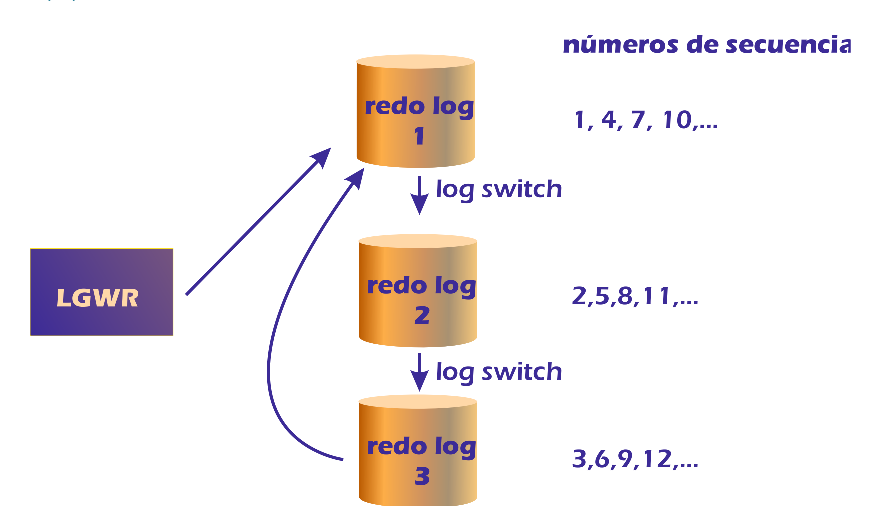


Ilustración 2, Funcionamiento de los archivos redo log

El evento **log switch** se puede provocar manualmente mediante la instrucción:

```
ALTER SYSTEM SWITCH LOGFILE;
```

multiplexación, grupos de redo log

Los archivos redo log suelen constituir grupos y así conseguir mediante **multiplexación**, copias de los archivos redo log. Cada grupo consta de varios archivos redo log clónicos correspondientes a las mismas secuencias y así si falla uno, tendremos otra copia disponible. Cuando se graba un dato de tipo redo, se graba en todos los archivos miembros del grupo.

Si no multiplexáramos los archivos, en un proceso de recuperación (por error grave), el sistema de bases de datos depende de la existencia de un solo archivo redo log (el que posee la última secuencia) para recuperar los datos. De ahí que no sólo haya que multiplexar, sino que los miembros de cada grupo deben estar en unidades diferentes para mayor seguridad.

Es tarea del administrador determinar la configuración de los redo log (número de grupos, número de archivos, ….). Así como decidir en qué discos se almacenan los grupos redo log, hay que tener en cuenta que la grabación de la información en los redo log debe de ser lo más rápida posible. La grabación en disco (desde el búfer) es crítica,

porque durante dicha grabación la sesión queda parada hasta terminar el volcado de datos.

Por defecto la base de datos tendrá tres grupos, pero con un solo miembro cada uno (es la situación mostrada en la Ilustración 2). Eso significa que no hay multiplexación y, por lo tanto, hay riesgo alto de perder un redo log.

vistas V$LOG y V$LOGFILE

Estas dos vistas permiten extraer información sobre los redo log actuales. **V$LOG** nos muestra una fila por cada grupo redo log.

Algunas columnas de V$LOG son:

- **GROUP#**. Indica el número de grupo.
- **SEQUENCE#**. Indica el número de secuencia.
- **BYTES**. Tamaño del archivo.
- **MEMBERS**. Número de miembros del grupo.
- **ARCHIVED**. Indica si es un redo log archivado(se explica posteriormente el significado)
- **STATUS**. Indica el estado del archivo: **UNUSED** (nunca se ha usado), **CURRENT** (secuencia en uso actualmente), **ACTIVE** (activo, aunque sea sin uso actual, esto significa que se requeriría en caso de recuperación de los datos), **CLEARING** (en inicialización), **CLEARING_CURRENT** (en inicialización desde un hilo cerrado, esto suele suponer un fallo en la creación), **INACTIVE** (sin uso, innecesario para la recuperación).
- **THREAD#**. Número de hilo, número de proceso LGWR que crea el log.
- **FIRST_CHANGE#**. Número de secuenciadle primer **SCN** grabado en el archivo (**System Change Number**, se explica más adelante) más bajo.
- **FIRST_TIME**. Fecha correspondiente al primer SCN comentado en el punto anterior.

De tal manera que la instrucción:

```
SELECT GROUP#, SEQUENCE#, MEMBERS, STATUS FROM V$FILE;
```

Nos mostraría lo más importante, grupos actuales, secuencias de los mismos y nº de miembros que tienen. Además de saber cuál es el grupo en uso en este momento.

V$LOGFILE nos permite indagar sobre los archivos redo log, de modo que nos muestra una fila por cada archivo redo log que tengamos. Lo que aparece en esta vista es:

- **GROUP#**. Indica el número de grupo.
- **STATUS**. Estado del archivo: **null** (está activo ahora), **INVALID** (archivo no accesible, error), **STALE** (archivo incoherente), **DELETED** (borrado, archivo sin uso)
- **TYPE**. **ONLINE** o **STANDBY** (disponible sin conexión).
- **MEMBER**. Ruta al archivo en disco.
- **IS_RECOVERY_DEST_FILE**. Indica si está en la zona de recuperación Flash.

añadir más grupos

Durante la creación de la base de datos si se usa el asistente (el programa **dbca**), en la versión avanzada de la instalación, se nos pregunta por los grupos que usaremos para los archivos redo y por los miembros que tendrán.

Si necesitamos hacerlo después, podemos (pero siempre es más problemático porque los archivos ya están en uso).

Para añadir más grupos se usa:

```
ALTER DATABASE ADD LOGFILE (listaDeArchivosMiembros) SIZE tamaño
[GROUP nº];
```

La lista de archivos miembros indica los miembros del grupo en cuestión. Se usa la ruta al archivo, si no se indica ruta se guardará en la ruta por defecto a la base de datos de Oracle. La cláusula **SIZE** nos permite indicar el tamaño del archivo (por ejemplo ***6M*** son seis Megabytes) y la opcional GROUP permite indicar el número del grupo; de no hacerlo se tomará como número el siguiente disponible.

Ejemplo:

```
ALTER DATABASE ADD LOGFILE
('c:\app\asir\oradata\asir\REDO04.LOG','d:\back\REDO004.LOG')
SIZE 6M;
```

En el ejemplo se crea un nuevo grupo REDO que ahora tendrá dos miembros multiplexados cada uno correspondiente a la ruta de archivo que se indica.

borrar grupos redo log

Para borrar un grupo se usa:

```
ALTER DATABASE DROP LOGFILE GROUP n
```

Donde ***n*** es el grupo a borrar. Por supuesto sólo se pueden borra grupos que actualmente no estén activos.

arreglar grupos corruptos

A veces un grupo tiene los datos corruptos, en cuyo caso conviene vaciarle e inicializarle de nuevo (durante el proceso el estado es CLEARING). Para iniciar un grupo se usa (si ***n*** es el grupo a inicializar):

```
ALTER DATABASE CLEAR LOGFILE GROUP n;
```

Esta instrucción no se puede realizar si hay sólo dos grupos redo log o si está en uso el grupo que se desea inicializar.

añadir miembros a un grupo

Para añadir nuevos miembros a un grupo se debe seguir esta sintaxis:

```
ALTER DATABASE ADD
LOGFILE MEMBER ruta TO GROUP n;
```

Por ejemplo:

```
ALTER DATABASE ADD LOGFILE MEMBER 'd:\back\REDO002.LOG'
TO GROUP 2
```

Nada más crear el miembro del grupo, el estado del archivo será **INVALID**, pero pasará a ser válido con su primer uso.

borrar un miembro a un grupo

Sólo se debe borrar para evitar problemas (archivos incoherentes o corruptos). No podremos eliminar miembros del grupo que esté en estado **CURRENT** o **ACTIVE** (el que se está usando ahora para volcar la caché) ni tampoco dejar un grupo sin miembros. Tampoco podremos borrar un miembro si el resto de miembros son inválidos.

Para eliminar un archivo redo log de un grupo se hace:

```
ALTER DATABASE DROP LOGFILE MEMBER ruta;
```

Hay que indicar simplemente la ruta al archivo redo y se quitará del grupo del que era miembro

mover de sitio archivos redo log

A veces necesitamos mover los archivos a otro sitio (como otra unidad de disco). Esto requiere dos pasos:

(1) Apagar la base de datos con **SHUTDOWN** (**NORMAL**, **IMMEDIATE** o **TRANSACTIONAL**).

(2) Desde el sistema operativo mover los archivos deseados a la nueva ubicación

(3) Arrancar la base de datos y montarla, pero no abrirla **STARTUP MOUNT**

(4) Modificar el nombre de los archivos para reflejar la nueva ubicación

```
ALTER DATABASE
RENAME FILE rutaAntiguaArchivo1 [, rutaAntiguaArchivo2 [,...]]
TO rutaNuevaArchivo1 [, rutaNuevaArchivo2 [,...]];
```

(5) Abrir la base de datos: **STARTUP OPEN;**

modos NOARCHIVELOG y ARCHIVELOG

En modo **NOARCHIVELOG**, los redo log se van llenando y en cuanto se llena uno, se pasa al siguiente grupo en el que se seguirán almacenando el resto de datos, y así sucesivamente. Pero, como sea explicado en el apartado anterior, si hemos llenado el último grupo, ocurrirá un nuevo evento **LOG SWITCH** y entonces los siguientes datos se sobrescriben sobre el primer archivo. Eso hace que se puedan perder datos en caso de que la instancia de Oracle se cierre por motivos imprevistos.

En modo **ARCHIVELOG**, a medida que se van llenando los datos de los archivos redo, éstos se copian a un archivo histórico (llamado ***redo archivado***) y así todos los datos redo están disponibles en todo momento. Es decir mantiene un histórico de todos los archivos redo log. Esa es la labor del proceso **ARC*n*** (en realidad son varios procesos, la ***n*** indica el número de grupo de archivo redo que actualiza; ARC1 es el encargado de escribir en el grupo 1, ARC2 el segundo grupo,...).

Al igual que ocurre con los archivos redo log normales, los archivados se pueden multiplexar para aumentar la seguridad de los mismos.

Lógicamente el problema de trabajar en modo ARCHIVELOG está en el alto consumo de disco que hacemos en este modo de trabajo.

Para saber en qué modo está la base de datos, se debe consultar la columna **LOG_MODE** de la vista **V$DATABASE** que contiene la información general de la base de datos:

```
SELECT LOG_MODE FROM V$DATABASE;
```

Aparecerá el texto **ARCHIVELOG** o **NOARCHIVELOG**.

Para cambiar de un modo a otro, la base de datos debe de estar en modo **mount**, es decir en estado de montada pero no abierta. Es el estado en el que se admite modificar el funcionamiento de la base de datos, pero en el que todavía no admiten usuarios. Esto es posible mediante las operaciones (desde una conexión de administrador):

```
SHUTDOWN IMMEDIATE /* Cierra la base de datos */
STARTUP MOUNT /* Monta la base de datos */
ALTER DATABASE ARCHIVELOG; /* cambia a modo ARCHIVELOG */
ALTER DATABASE OPEN; /* La base de datos pasa a estar abierta */
```

Cuando estamos en modo ARCHIVE LOG, el proceso **ARC*n***, donde la ***n*** es el número de secuencia de archivo es el encargado de ir archivando los datos a medida que llenamos los archivos LOG en memoria.

consultar los archivados redo log

La vista **V$ARCHIVED_LOG** nos muestra información sobre los redo log archivados. Fundamentalmente el nombre del archivo (ruta completa al mismo) mediante la columna **NAME** y el número de secuencia redo log que se almacena en dicho archivo (**#SEQUENCE**)

modificar la ruta y nombre de los archivados redo

La ruta en la que se almacenan los archivos redo log históricos se puede mostrar con:

```
SELECT * FROM V$PARAMETER
WHERE NAME LIKE 'log_archive_dest%';
```

O con:

```
SHOW PARAMETER log_archive_dest;
```

Mostrará las localizaciones en las que se guardan los históricos (puede ser más de una, de ser así se almacenan en paralelo). Si la ruta vale nula, entonces se guarda en la ruta predeterminada (que suele ser ***ORACLE_BASE***\FLASH_RECOVERY_AREA\ARCHIVELOG).

Para variar la ruta en la que se almacenan los archivos y tenerla más controlada, necesitamos modificar el parámetro **LOG_ARCHIVE_DEST_1**, en el caso, habitual, en el que nuestros históricos se almacenen sólo en una ubicación: de otro modo habría que modificar todos los parámetros LOG_ARCHIVE_DEST necesarios y de ese modo se multiplexarían los ficheros.

Para modificar la ruta se usa el comando:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='location=/nuevaRuta'
SCOPE=BOTH;
```

De esa forma podemos controlar mejor dónde se guardan los históricos. El nombre de los archivos se puede calibrar por ejemplo con:

```
ALTER SYSTEM SET
LOG_ARCHIVE_FORMAT='%t_%s_%r.arc' SCOPE=SPFILE;
```

El nombre de los archivos será el número de grupo redo log (***%t***) al que se refiere el archivo, el número de secuencia (***%s***) y finalmente el número **resetlog** (***%r***), el cual que asegura que el resto del nombre del archivo es único.

archivos de control

Se trata de archivos binarios y de tamaño pequeño que contienen la estructura de la base de datos:

- Nombre de la base de datos
- Fecha y hora de la creación de la base de datos
- Información sobre **checkpoints** y **redo logs**.
- Modo de archivado de la base de datos.
- Número de secuencia del redo log actual.
- Metadatos para backup y recuperación de datos.

En definitiva almacenan toda la información de la estructura de la base de datos. Siempre hay un archivo de control, pero como su coherencia es crítica para la base de datos, se recomienda encarecidamente multiplexar en al menos dos archivos de control para tener siempre una copia al día (se pueden multiplexar en más).

Se pueden consultar los archivos de control en uso mediante:

```
SELECT * FROM V$CONTROLFILE;
```

Las secciones que se almacenan en los archivos de control también se pueden consultar mediante:

```
SELECT * FROM V$CONTROLFILE_RECORD_SECTION;
```

Los archivos de control se crean durante la creación en sí de la base de datos. Normalmente Oracle crea dos archivos de control multiplexados, pero lo ideal además es que cada uno se almacene incluso en un disco diferente.

La lista de los archivos de control se configura mediante el parámetro **CONTROL_FILES**. Se puede consultar mediante:

```
SHOW PARAMETER CONTROL_FILES;
```

O mediante:

```
SELECT * FROM V$PARAMETER WHERE NAME='control_files';
```

En ambos casos, podemos ver que el valor del parámetro es una lista de las rutas a cada copia del archivo de control separadas por comas.

añadir o modificar archivos de control

Si deseamos añadir una nueva copia multiplexada de los archivos de control, deberemos modificar el parámetro CONTROL_FILES en el archivo de parámetros y después copiar el archivo de control a la nueva ubicación deseada.

Esa copia hay que hacerla con la base de datos apagada para asegurar que el archivo de control es coherente. Por ejemplo si deseamos añadir un nuevo archivo de control para tener una copia multiplexada más:

(1) Comprobar la ubicación de los archivos de control actuales:

```
SELECT VALUE FROM V$PARAMETER WHERE NAME='control_files';
```

Suponer que sale la información:

```
C:\APP\ASIR\ORADATA\ASIR\CONTROL01.CTL,
C:\APP\ASIR\FLASH_RECOVERY_AREA\ASIR\CONTROL02.CTL
```

(2) Modificar ese parámetro para indicar cuáles serán los nuevos archivos de control

```
ALTER SYSTEM SET
CONTROL_FILES='C:\APP\ASIR\ORADATA\ASIR\CONTROL01.CTL',
'C:\APP\ASIR\FLASH_RECOVERY_AREA\ASIR\CONTROL02.CTL',
'd:\back\control03.ctl' scope=SPFILE;
```

De esta forma se ha añadido el archivo *control03*.ctl a la lista de archivos de control.

(3) Apagar la base de datos en modo normal (si no hay más remedio en modo IMMEDIATE, pero debería ser nuestra última opción).

```
SHUTDOWN NORMAL;
```

(4) En el sistema operativo copiar uno de los archivos de control existentes a la nueva ubicación dándole el nombre indicado en el paso **2. Es muy importante que el nombre del archivo copia sea exactamente el indicado en el parámetro CONTROL_FILES**

(5) Abrir la base de datos

```
STARTUP OPEN;
```

archivos de parámetros

Comentados en el tema uno. Contienen los parámetros generales de configuración de la base de datos. Se encuentra por defecto en la carpeta de la base de datos (en Windows, normalmente dentro de *ORACLE_HOME* en la carpeta **database**, en Linux sería la carpeta **dbs**).

Normalmente Oracle utiliza un archivo de parámetros binario, conocido con las siglas SPFILE. Podemos hacer copia de él en un archivo de texto usando la instrucción.

```
CREATE PFILE[=ruta] FROM SPFILE[=ruta];
```

Sin indicar ruta, se crean los archivos usando la ruta por defecto.

Para hacer que Oracle use un archivo PFILE en lugar del predeterminado se usa la siguiente instrucción para arrancar Oracle:

```
STARTUP PFILE=ruta
```

archivos de contraseñas

Contienen la lista de contraseñas cifradas, en caso de que ésta sea la forma de autentificar usuarios (según lo comentado en el tema uno). Todos los usuarios administradores tienen en ese archivo las contraseñas, por lo que interesa tener copia del mismo a menudo.

Se crean mediante la utilidad **orapwd** que se instala junto a Oracle. Por defecto los archivos de contraseñas están en la carpeta de la base de datos (al igual que los archivos de parámetros).

(3.2.2) posibilidades de copias de seguridad en Oracle

Oracle dispone de tres posibilidades principales para recuperar información:

(1) **La herramienta RMAN**. Es la opción preferente por su potencia y posibilidades. En estos apuntes no se comenta debido a su complejidad, pero indudablemente es la mejor opción.

(2) **Hacer copias manuales**. Se tratan a partir del siguiente apartado. Se trata de utilizar los comandos del sistema para realizar copias. Permiten reconocer el funcionamiento de Oracle y, bien utilizadas, son una opción muy rápida (especialmente si se automatiza el proceso).

(3) **Recuperaciones de tipo flashback**. Que permiten retroceder a un punto concreto en el tiempo en el que se solucionan los errores de usuario cometidos (es una recuperación especialmente pensada para esos problemas).

(3.3) copias físicas en frío de Oracle

(3.3.1) realización de la copia de seguridad en frío en modo NOARCHIVELOG

Permite realizar copia exacta de los datos de Oracle cuando ésta se ha desconectado. En este modo necesitamos copiar:

- Archivos de datos
- Archivos de control
- Archivos redo log, aunque no es imprescindible ya que si hemos cerrado la base de datos sin usar una forma abrupta (como SHUTDOWN ABORT) se habrá producido un checkpoint, lo cierto es que si se hace copia de ellos la restauración podría consistir en copiar todo como estaba y la base de datos arrancará sin más.

- Archivos de parámetros
- Archivos de contraseñas, sólo si se utilizan

La cuestión es saber dónde están y para ello se usan estos comandos:

```
SELECT name FROM V$DATAFILE; /* Archivos de datos */
SELECT name FROM V$CONTROLFILE; /* Archivos de control */
SELECT member FROM V$LOGFILE; /* Archivos redo */
SELECT name FROM V$TEMPFILE; /* Temporales*/
SELECT name,value FROM V$PARAMETER
WHERE name='spfile'; /* SPFILE */
```

Para ligar la información de los archivos de datos respecto a los tablespaces con los que están relacionados, se usa la vista:

```
SELECT t.name "Tablespace", f.name "Datafile"
FROM V$TABLESPACE t, V$DATAFILE f
WHERE t.ts# = f.ts#
ORDER BY t.name;
```

Gracias a esos comandos se puede obtener la lista completa de archivos necesarios. Con esa información, los pasos son:

(1) Obtener la lista de archivos, usando los SELECT anteriores, aunque mejor aún es ejecutar estas instrucciones:

```
SPOOL 'rutaAUnArchivo.BAT'
SELECT 'copy ' || name || ' rutaCarpetaBackup' FROM V$DATAFILE
UNION
SELECT 'copy ' || name || ' rutaCarpetaBackup' FROM V$CONTROLFILE
UNION
SELECT 'copy ' || name || ' rutaCarpetaBackup' FROM V$TEMPFILE
UNION
SELECT 'copy ' || member || ' rutaCarpetaBackup' FROM V$LOGFILE
UNION
SELECT 'copy' ||  value || ' rutaCarpetaBackup'  FROM V$PARAMETER
WHERE name='spfile';

SPOOL OFF
```

El comando **Sql*Plus spool**, permite volcar a un archivo, las instrucciones y los resultados. De ese modo en el archivo **.bat** generado ya estarán escritas las instrucciones **copy** que permiten hacer copia de los archivos de Oracle.

Además hay que ejecutar el comando:

```
CREATE PFILE='rutaNuevoPFILE' FROM SPFILE; /* Copia los parámetros
a un nuevo archivo de tipo PFILE */
```

Que crea un archivo de parámetros a partir del archivo **spfile** de parámetros actual. Ese archivo sólo será necesario usarle en caso de que se pierda o falle el archivo de parámetros original. En ese caso se ejecutaría el comando contrario para generar el **spfile** a partir del **pfile** creado anteriormente.

(2) Ejecutar el comando **SHUTDOWN IMMEDIATE**. Se cerrará la instancia de Oracle en buenas condiciones para hacer una copia en frío.

(3) Copiar los archivos desde el sistema Operativo. Si hemos creado un script para hacer dicha copia, ejecutarle.

(4) Ejecutar el comando **STARTUP**.

restauración con archivos redo log

Si disponemos de archivos redo log y les queremos usar para dejar la base de datos tal cual estaba en el momento de la copia, bastaría seguir estos pasos:

(1) Apagar la instancia (incluso con **SHUTDOWN ABORT**)

(2) Copiar los archivos desde el sitio de copia al original

(3) Arrancar la base de datos con **STARTUP**.

La base de datos arrancará sin problemas y estará en el mismo estado que anteriormente.

Esta forma de restaurar se usa cuando no hay más remedio ya que perderemos todos los cambios realizados desde el momento en el que hicimos la copia de seguridad

restauración sin archivos redo log

El modo de recuperación anterior significa un retorno al momento en el que hicimos la copia, funciona de forma segura pero tiene el problema de que perderemos los cambios entre el momento de la copia y el momento en el que tuvimos que restaurarla.

En el modo NOARCHIVELOG es posible restaurar aún sin los redo log, esto intenta restaurar la base de datos para dejarla en el último estado grabado en los redo log. Es decir pretende que los últimos cambios (si están en los redo log) pasen a la base de datos. Pasos:

(1) Apagar la instancia (incluso con **SHUTDOWN ABORT**)

(2) Copiar los archivos de datos y de control a su ubicación original

(3) Montar la instancia, pero no abrirla:

```
STARTUP MOUNT
```

(4) Ejecutar el siguiente comando puesto que no disponemos de redo log:

```
RECOVER DATABASE UNTIL CANCEL;
```

(5) Abrir la base de datos indicando que hay que resetear los redo log:

```
ALTER DATABASE OPEN RESETLOGS
```

Si fallara la restauración de esta forma, deberíamos intentar recuperar la base de datos usando los archivos redo log de la copia de seguridad.

(3.3.2) realización de la copia de seguridad en frío en modo ARCHIVELOG

En este caso la idea es restaurar la base de datos al punto en el que se hizo la copia y a partir de ahí ejecutar transacciones hasta recuperar completamente la base de datos. En este tipo de restauración, no se deben incluir los archivos redo normales al restaurar los archivos. La razón es que se sobrescribirían con la información de los archivos redo copiados y no recuperaríamos la base de datos al último estado (ya que los archivos redo en modo CURRENT o ACTIVE contienen información fuera de los archivos de datos).

Los pasos para realizar la copia son idénticos a los comentados en el punto anterior, salvo en el hecho de que no se hace copia de los archivos redo. Al tener la información en los ARCHIVELOG, es posible retroceder mucho más tiempo y, por lo tanto recuperar la base de datos.

La restauración se realiza de la misma forma que cuando se desea realizar copia en caliente de la base de datos.

(3.4) copias en caliente de la base de datos

requieren el uso de archivos redo archivados (modo ARCHIVELOG).

(3.4.1) mostrar los redo log archivados

En este caso la base de datos debe de estar en modo **ARCHIVELOG**, de otro modo no es posible hacer copias en caliente porque la recuperación no sería posible al faltar datos para realizarla. Los archivados permite almacenar toda la información anterior y eso es indispensable para este tipo de copia.

Los redo log archivados se pueden listar desde este comando:

```
SELECT THREAD#, SEQUENCE#, NAME
FROM V$ARCHIVED_LOG;
```

La vista **V$ARCHIVED_LOG** contiene todos los datos de los archivos redo log históricos, entre ellos **THREAD#** (nº de grupo), **SEQUENCE#** (nº de secuencia) y **NAME** (ruta completa al archivo)

(3.4.2) pasos para realizar copia completa en caliente de la base de datos

(1) Asegurar que estamos en modo ARCHIVELOG. Ejecutando el comando:

```
ARCHIVE LOG LIST;
```

Podría obtener el resultado:

Modo log de la base de datos	Modo de Archivado
Archivado automático	Activado
Destino del archivo	C:\app\asir\arch
Secuencia de log en línea más antigua	1
Siguiente secuencia de log para archivar	2

Secuencia de log actual	2

Lo cual nos ofrece toda la información para verificar que estamos en modo ARCHIVELOG y en qué momento del archivado de los redo log estamos.

(2) Obtener el nombre de los archivos de datos y su relación con los tablespaces:

```
SELECT TABLESPACE_NAME, FILE_NAME
FROM DBA_DATA_FILES ORDER BY TABLESPACE_NAME;
```

(3) Obtener los datos de la secuencia actual de los redo log, ya que como mínimo para restaurar la base de datos al menos necesitamos los redo log generados durante la operación de backup.

```
SELECT THREAD#, MAX(SEQUENCE#)
FROM V$LOG GROUP BY THREAD#
ORDER BY THREAD#;
```

(4) Colocar la base de datos en estado de backup:

```
ALTER DATABASE BEGIN BACKUP;
```

Esta instrucción coloca todos los tablespaces de la base de datos en estado de backup, y eso significa que no será posible actualizarlos durante la operación. Esto puede ser muy peligroso en bases de datos que se actualizan a menudo.

La razón de colocar la base de datos en ese estado es para poder copiar los archivos de datos, sin riesgo de que desde el inicio hasta el fin de la copia hayan cambiado. Como esto puede ser una tarea muy larga, se aconseja a paralizar individualmente cada tablespace, copiar sus archivos de datos y después reanudar el tablespace y pasar al siguiente. Dejar a un tablespace en modo backup se hace con:

```
ALTER TABLESPACE nombreTablespace BEGIN BACKUP;
```

Para comprobar qué archivos están en estado de backup, se puede consultar con:

```
SELECT * FROM V$BACKUP;
```

En esta vista dinámica, aparece en estado **ACTIVE** los ficheros en estado de backup. También con:

```
SELECT NAME, STATUS, FUZZY FROM V$DATAFILE_HEADER;
```

Esta vista muestra el nombre de cada archivo de datos, su situación (normalmente **ONLINE**) y si está en estado de backup (**FUZZY**=yes)

(5) Copiar los archivos de datos (desde el sistema operativo). Si hemos dejado sólo a un tablespace en estado de backup, entonces copiar los archivos de datos relacionados con dicho tablespace.

(6) Salir del estado de backup. En el caso de haber detenido la base de datos entera con:

```
ALTER DATABASE END BACKUP;
```

Si hemos detenido sólo un tablespace, entonces se reanuda con:

```
ALTER TABLESPACE nombreTablespace END BACKUP;
```

A continuación pasaríamos de nuevo al paso cuarto para continuar con el siguiente tablespace.

(7) Archivar el redo log actual para asegurar que la operación de fin de backup queda grabada en él:

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

(8) Consultar el número actual de redo log para saber cuáles han cambiado en todo el proceso.

```
SELECT THREAD#, MAX(SEQUENCE#)
FROM V$LOG GROUP BY THREAD#
ORDER BY THREAD#;
```

Puesto que la copia es en caliente, no es posible pararla.

(9) Hacer copia del archivo de control:

```
ALTER DATABASE BACKUP CONTROLFILE TO 'rutaBackup.ctl' REUSE;
```

La cláusula **REUSE** permite sobrescribir la copia de seguridad del archivo de control anterior si existiera. También se le puede copiar mediante el comando:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

Esta instrucción genera un script que al ejecutarse sería capaz de generar el archivo de control. El archivo se almacena donde diga el parámetro **USER_DUMP_DEST**, modificable con:

```
ALTER SYSTEM SET USER_DUMP_DEST="nuevaRutaAlDirectorio"
```

(10) Copiar los archivos redo log generados durante la copia de seguridad.

(11) Copiar archivo de parámetros

```
CREATE PFILE = 'rutaArchivo.ora' FROM SPFILE;
```

(3.4.3) verificación de las copias

Los archivos de datos son fundamentales, por ello una buena idea es verificar que realmente las copias son correctas. Para ello Oracle admite el uso de la herramienta **dbv** que se ejecuta desde el sistema operativo. Ejemplo:

```
dbv file=c:\backup2\users01.dbf logfile=c:\backup2\dbv.log
```

En el ejemplo se verifica el archivo de datos ***users01.dbf*** y el resultado del análisis se guarda en el archivo ***dbv.log*** (ambos en la carpeta ***backup2*** del disco duro). Ejemplo de resultado en el archivo log:

```
DBVERIFY: Release 11.2.0.1.0 - Production on Vie Feb 3 01:08:06 2012

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


DBVERIFY - Iniciando verificación : FILE = C:\BACKUP2\USERS01.DBF

DBVERIFY - Verificación terminada

Total de Páginas Examinadas     : 52960
Total de Páginas Procesadas (Datos): 76
Total de Páginas con Fallos (Datos): 0
Total de Páginas Procesadas (Índice): 3718
Total de Páginas con Fallos (Índice): 0
Total de Páginas Procesadas (Otras): 19797
Total de Páginas Procesadas (Seg): 0
Total de Páginas con Fallos (Seg): 0
Total de Páginas Vacías         : 29369
Total de Páginas Marcadas como Corruptas: 0
Total de Páginas de Entrada     : 0
Total de Páginas Cifradas       : 0
SCN de Bloque Superior          : 2171581 (0.2171581)
```

(3.4.4) restauración completa de una copia en caliente

restauración completa apagando la base de datos

Se utiliza cuando hay un archivo de datos que no funciona. En ese caso la restauración es completa si disponemos de los archivos redo log archivados. Los pasos para la recuperación son:

(1) Colocar la base de datos en estado **MOUNT**. Por ejemplo:

```
SHUTDOWN ABORT; //simula fallo del sistema
STARTUP MOUNT;
```

(2) Copiar el archivo (desde el sistema operativo) dañado desde la carpeta del backup. El archivo copiado es anterior al que existía por lo que la marca temporal de dicho archivo (llamada **SCN**) es anterior y eso implica que no se puede abrir la base de datos todavía.

(3) Ejecutar la instrucción **RECOVER** correspondiente. Dependiendo del fallo puede ser: **RECOVER DATABASE** (recupera todos los archivos de la base de datos), **RECOVER TABLESPACE** (recupera todos los archivos de un tablespace), **RECOVER DATAFILE** (recupera un archivo de datos). Ejemplo:

```
RECOVER DATAFILE 4;
```

Esa instrucción recupera el archivo de datos número 4 (**se podría utilizar la ruta completa, en lugar del número**). Para saber los archivos que se deben recuperar, se puede consultar (antes de ejecutar la instrucción) con:

```
SELECT * FROM V$RECOVER_FILE;
```

Si deseamos recuperar todo el tablespace sería, por ejemplo:

```
RECOVER TABLESPACE USERS;
```

Al recuperar, lo normal es que Oracle escriba este aviso:

```
ORA-00279: el cambio 2395190 generado en 02/03/2012 00:52:37 es necesario
para el thread 1
ORA-00289: sugerencia: C:\APP\ASIR\ARCH\1_3_774214223.ARC
ORA-00280: el cambio 2395190 para el thread 1 estß en la secuencia n·mero 3


Especificar log: {<RET>=sugerido | nombre_archivo | AUTO | CANCEL}
```

Referido a que necesitamos los archivos redo archivados para recuperar coherentemente los datos. Normalmente basta con utilizar la opción sugerida (es decir pulsar **Intro**) para recuperar. Pero podemos especificar un archivo redo log archivado propio (de nuestra copia de seguridad) indicando el nombre del archivo ante la pregunta anterior. De ahí la importancia de tener archivados los redo log archivados de la copia de seguridad.

(4) Abrir la base de datos:

```
ALTER DATABASE OPEN;
```

restauración completa manteniendo la base de datos online

Si no podemos detener la base de datos, porque tiene que estar permanentemente online, entonces una vez que detectemos el archivo que ha fallado, los pasos serían:

(1) Ejecutar la orden:

```
ALTER DATABASE DATAFILE 'rutaArchivoQueFalla' OFFLINE;
```

Con esto, sólo ese archivo se queda **offline**.

(2) Copiar el archivo de la copia de seguridad

(3) Ejecutar la instrucción:

```
RECOVER DATAFILE nombreArchivoÓNúmeroDeArchivo;
```

Tocará arreglar el asunto de los redo log archivados según lo comentado en el punto anterior

(4) Finalmente:

```
ALTER DATABASE DATAFILE 'rutaArchivoQueFalla' ONLINE;
```

restaurar archivos de control

Si ocurre un fallo a un archivo de control y tenemos copias multiplexadas del mismo (como se comentó en el apartado añadir o modificar archivos de control, página 19), bastará apagar la base de datos y copiar el archivo correcto sobrescribiendo el incorrecto. Al iniciar todo debería funcionar correctamente.

Lo malo es si es un fallo general de todos los archivos de control. Ahora los pasos son:

(1) Parar la base de datos

(2) Copiar el archivo de control desde la carpeta de copia de seguridad

(3) Copiar **todos** los archivos de datos desde la copia de seguridad

(4) Montar la base de datos (STARTUP MOUNT)

(5) Ejecutar la orden:

```
RECOVER DATABASE USING BACKUP CONTROLFILE;
```

Eso restaura la situación desde el archivo de control. Oracle sabe qué archivo de control restaurar porque sabe como se hizo la copia.

(6) Nuevamente tras la orden anterior, Oracle nos interrogará por los redo log; deberemos resolver la situación al igual que en los apartados anteriores.

(7) Abrir mediante:

```
ALTER DATABASE OPEN RESETLOGS;
```

(3.4.5) restauración incompleta

Este tipo de restauración se utilizan cuando tenemos problemas con la copia de seguridad. Con este tipo de recuperación volvemos a un instante concreto del tiempo ya que no podremos volver al último punto de la base de daros. En ese caso, ya no podemos recuperar completamente la base de datos, pero disponemos de estas posibilidades:

- Retroceder hasta el momento en el que la base de datos detectó por última vez que funcionaba correctamente. Lo cual se hace con:

```
RECOVER DATABASE UNTIL CANCEL;
```

- Retroceder a un número de SCN concreto:

```
RECOVER DATABASE UNTIL CHANGE 123412;
```

- Retroceder a un momento en el tiempo concreto:

```
RECOVER DATABASE UNTIL TIME '2011-11-17:12:00:00';
```

Los pasos completos para este tipo de recuperación serían:

(1) Apagar la base de datos (SHUTDOWN ABORT)

(2) Copiar todos los archivos de datos desde la copia de seguridad

(3) Montar la base de datos (STARTUP MOUNT)

(4) Ejecutar la instrucción **RECOVER DATABASE** más apropiada, según lo comentado anteriormente. Oracle preguntará por los redo log, en este caso indicamos la opción **CANCEL** para detener el proceso de restauración.

(5) Abrir la base de datos con:

```
ALTER DATABASE OPEN RESETLOGS;
```

Perderemos los cambios a partir de ese punto en el tiempo.

(6) Es conveniente hacer una copia en frío de la base de datos.

(3.5) recuperaciones de tipo Flashback

En las últimas versiones de Oracle se ha incluido una tecnología que permite arreglar los errores de usuario sin necesidad de utilizar copias de seguridad. Son las recuperaciones de tipo Flashback. Consisten en hacer que toda la base de datos (o una tabla) regrese a un punto anterior en el tiempo, en el que los datos eran correctos.

Arregla los problemas que causa un usuario que elimina definitivamente datos y luego se arrepienta (con un **DROP TABLE** por ejemplo), bases de datos que no arrancan tras hacer cambios en la configuración o bien restauraciones a partir de copias de seguridad que no funcionan porque faltan archivos.

(3.5.1) área rápida de recuperación

Para que la posibilidad de una recuperación de una base de datos de tipo Flashback funcione, se necesita que funcione lo que Oracle denomina el **Flash Recover Area** o simplemente **FRA** (**área rápida de recuperación**). Se trata de un espacio en disco que se dedica a almacenar los archivos necesarios que contengan la información necesaria para poder retroceder la base de datos a un punto seguro en el tiempo.

En esa área se pueden almacenar copias del archivo de control, archivos redo log, históricos de archivos redo log, archivos log para el flashback,... Se puede establecer una cuto a fin de que dicho área no tome un tamaño excesivo.

Para que estas recuperaciones funciones hay que calibrar estos parámetros:

- **DB_RECOVERY_FILE_DEST_SIZE**. Tamaño que dejamos para el FRA. Se puede modificar en el archivo PFILE (si ese es el tipo de archivo de parámetros que usamos) o con ALTER SYSTEM SET. Ejemplo:

```
ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE=5G
SCOPE=BOTH;
```

- **DB_RECOVERY_FILE_DEST**. Ruta en la que se guardan los archivos flashback de recuperación. Se puede modificar e indicar una ubicación personal. Es recomendable que el FRA se encuentre en una unidad de disco distinta a la de la base de datos.
- **DB_FLASHBACK_RETENTION_TARGET**. Número máximo de minutos que podrá retroceder la operación flashback de la base de datos. Es decir, como mucho a qué instante del tiempo, contado en minutos partir de ahora, podremos volver.

Por otro lado para activar el uso de flashback, se usa:

```
ALTER DATABASE FLASHBACK ON;
```

Si en lugar de **ON**, indicamos **OFF**, entonces desactivamos esta posibilidad. Además FRA funciona si estamos en modo **ARCHIVELOG**.

Para comprobar que estamos usando FRA, se usa la instrucción:

```
ARCHIVE LOG LIST
```

Que, si estamos usando FRA, obtiene algo como esto:

```
Database log mode Archive Mode
Automatic archival Enabled
Archive destination USE_DB_RECOVERY_FILE_DEST
```

Pudiera ocurrir que los históricos redo log no se estén almacenando en la zona FRA. Por ello conviene comprobar el parámetro **LOG_ARCHIVE_DEST**:

```
SHOW PARAMETER LOG_ARCHIVE_DEST;
```

Veremos la lista de ubicaciones donde se guardan los archivados redo log. Si deseamos almacenar por ejemplo en el FRA y una copia guardarla en otro sitio. Podremos hacer, por ejemplo:

```
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_1='location=USE_DB_RECOVERY_FILE_DEST';
ALTER SYSTEM SET
LOG_ARCHIVE_DEST_2='location=/app/oracle11g/arch;
```

En ese ejemplo se guardan los históricos redo log en el FRA y en la ruta ***/app/oracle11g/arch***. Hay que decir que los archivos redo log archivados en el FRA no usan el formato de nombre de archivo especificado en el parámetro LOG_ARCHIVE_FORMAT; de otro modo Oracle no podría automatizar las operaciones de Flashback.

(3.5.2) SCN

El SCN (***system change number***) es un número que asigna Oracle a cambio no anulable mediante **ROLLBACK** en la base de datos. Para consultar cuál es el actual, se usa:

```
SELECT CURRENT_SCN
FROM V$DATABASE;
```

Esta instrucción devuelve el SCN actual.

(3.5.3) parámetro UNDO_RETENTION

Este parámetro permite especificar el tiempo máximo que se admitirá para deshacer los cambios definitivos. Para estas operaciones que deshacen acciones se unas un tablespace con los datos ***undo*** (deshacer) necesarios. A través de este parámetro podemos saber y cambiar el tiempo máximo permitido para estas acciones. Lógicamente más tiempo, más espacio se necesita.

(3.5.4) recuperaciones de tablas mediante flashback

prerrequisitos

Un usuario que desee hacer un flashback para recuperar el estado anterior de una tabla debe cumplir lo siguiente:

- Tener asignado el privilegio **FLASHBACK ANY TABLE** o el permiso **FLASHBACK** sobre la tabla que se desea recuperar.
- Privilegios **SELECT, INSERT,DELETE**, **UPDATE** y **ALTER** sobre la tabla.
- Para usar un punto de restauración, además se debe tener asignado el rol **SELECT_CATALOG_ROLE** o el permiso **SELECT ANY DICTIONARY**.
- Las tablas a recuperar deben tener activado el **ROW MOVEMENT**, que permite movimiento por filas avanzado. Es posible activarlo con:

```
ALTER TABLE table ENABLE ROW MOVEMENT;
```

- Hay que activar la papelera de reciclaje. Se puede usar el parámetro **RECYCLEBIN** para hacerlo.
- Debe de estar activada la posibilidad de flashback sobre la base de datos. Posible con:

```
ALTER DATABASE FLASHBACK ON;
```

- También debe de estar activado en el tablespace al que pertenece la tabla (normalmente si el de la base de datos está, lo estará el del tablespace). Si no lo está se consigue con:

```
ALTER TABLESPACE nombre FLASHBACK ON;
```

restaurar una tabla hasta antes de su borrado (FLASHBACK BEFORE DROP)

Necesitamos que la papelera esté activada. Cuando se borra una tabla si se permite hacer flashback y la papelera está activada, la podremos ver en la papelera, mediante:

```
SELECT OBJECT_NAME, ORIGINAL_NAME, TYPE
FROM RECYCLEBIN;
```

Se mostrarán todos los elementos borrados. Para recuperar la tabla borrada se usa:

```
FLASHBACK TABLE tabla TO BEFORE DROP;
```

La tabla regresará. Lo malo es que algunas restricciones (y especialmente los índices) habrán cambiado de nombre. Deberemos renombrarlas, si queremos tenerles controlados.

restaurar una tabla a un SCN anterior

Se usa poco, porque no se suelen anotar los SCN. Pero funcionaría de esta forma:

```
SELECT CURRENT_SCN FROM V$DATABASE; --devuelve 76872341
.... //se hacenb otras operaciones
FLASHBACK TABLE tabla TO SCN 76872341;
```

Sólo es posible regresar si tenemos el ROW MOVEMENT de la tabla, activado.

restaurar una tabla a un momento concreto en el tiempo

Ejemplo:

```
FLASHBACK TABLE tabla TO TIMESTAMP
TO_TIMESTAMP('05/02/2012 1:54','dd/mm/yyyy hh24:mi');
```

Deja la tabla con la información que tenía a la 1:55 del 5/2/2012. Tras el apartado **TO_TIMESTAMP** basta con poner un dato válido de tipo **timestamp**. Otro ejemplo:

restaurar a una tabla a un punto de restauración

Un punto de restauración, es una marca que se hace y que luego permite volver al estado que la tabla tenía cuando se puso la marca. Proceso:

```
CREATE RESTORE POINT punto1;
... //operaciones
FLASH BACK TABLE table TO RESTORE POINT punto1.
```

(3.5.5) recuperaciones de bases de datos mediante flashback

prerrequisitos

- La base de datos debe de estar en modo **ARCHIVELOG**
- Está activado el área rápida de recuperación (FRA)
- Está activada la posibilidad de flashback de la base de datos (**ALTER DATABASE FLASHBACK ON**);

información sobre la base de datos flashback

La vista dinámica **V$FLASHBACK_DATABASE_LOG** nos muestra información sobre el momento más antiguo de flashback (incluido el SCN más antiguo), así como el tamaño de los archivos para el flashback.

Esta otra instrucción:

```
SELECT NAME,SCN,TIME,GUARANTEE_FLASHBACK_DATABASE
FROM V$RESTORE_POINT;
```

Muestra el nombre, el SCN y la fecha y hora de cada punto de restauración. La columna **GUARANTEE_FLASHBACK_DATABASE**, indica con **Y**, que dicho punto de restauración se puede utilizar con garantías para un flashback de base de datos.

EL SCN actual, se consulta con:

```
SELECT CURRENT_SCN
FROM V$DATABASE;
```

restaurar la base de datos

Las restauraciones de la base de datos se hacen igual que las de tabla, cambiando en la sintaxis **TABLE** y el nombre de la tabla, por la palabra **DATABASE**. Además se deben realizar con la base de datos en modo **MOUNT**, por lo que sólo se utilizan ante hechos muy graves.

Ejemplo:

```
SELECT CURRENT_SCN FROM V$DATABASE; //devuelve 2029587
.... //se hacen más operaciones y nos damos cuenta de un error en ellas
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
FLASHBACK DATABASE TO SCN 2029587
```

La base de datos se restaura. El problema ahora es que puede haber serios problemas, por ello primero hay que comprobar los datos y eso es mejor hacerlo abriendo la base de datos en modo sólo lectura:

```
ALTER DATABASE OPEN READ ONLY;
...//Comprobamos los cambios
```

Si todo va bien ahora hay que abrir de esta forma:

```
SHUTDOWN IMMEDIATE
STARTUP MOUNT
ALTER DATABASE OPEN RESETLOGS;
```

La última línea abre la base de datos poniendo al día los redo log y eso implica no poder ahora recuperar las posiciones de base de datos posteriores al momento del flashback.

Si al comprobar en modo sólo lectura, encontramos que el cambio no es el bueno; deberemos anular el flashback de esta forma:

- Haciendo otro flashback a otro SCN si queremos retroceder aún más
- Ejecutar esta orden para ir a un SCN posterior. Por ejemplo

```
RECOVER DATABASE UNTIL SCN 23781;
```

- Podemos anular completamente todo lo realizado por el flashback mediante:

```
RECOVER DATABASE;
```